Job Management System

The goal of the assignment is to create a system that will connect job seekers and employers. For this, it necessary to perform the following tasks:

User Registration:

- Develop a user registration feature that collects the following data: first name, last name, username, email, and password.
- For the employer role, include additional fields: isEmployer (a checkbox indicating if the user is an employer) and Company (the name of the company).
- Implement validation checks to ensure that all required fields are filled in. Display appropriate error messages if any required field is missing.
- Validate the email field to ensure it is in the correct format (e.g., "example@example.com"). Display an error message if the email is not in the correct format.
- Store the user registration details in a database. Design and implement the necessary database schema to store the user information securely.

User Roles:

- Implement a role-based system with two roles: "Employer" and "Applicant."
- If the user selects the employer role during registration (isEmployer is true), display additional fields related to the employer role, such as the Company field. Make the Company field mandatory for employers.
- If the user does not select the employer role, assign the applicant role to the user by default.
- Store the user's role information in the database along with other registration details.
- Implement appropriate security measures, such as password hashing and encryption, to protect user data.

Authorization:

- Implement an authentication feature that allows registered users to log in using their email and password.
- Create a login method where users can enter their email and password for authentication.
- Validate the user's credentials by verifying them against the stored data in the database. Compare the entered email and password with the corresponding values in the database.
- Implement a secure session management mechanism to maintain user login sessions. This mechanism should generate a unique session token or identifier upon successful login and associate it with the user's account.

- Use secure techniques such as JSON Web Tokens (JWT) to protect the session token from unauthorized access or tampering.
- Implement session expiration or timeout mechanisms to automatically log out users after a period of inactivity to enhance security.
- Provide appropriate error handling and feedback to users in case of login failures, such as displaying error messages for incorrect credentials or locked accounts.

Pre-Filled Data:

Create Database Tables with Pre-Filled Data for User Selection:

- Skills: Create a database table named "skills" to store a list of skills applicable to user profiles and CVs. Define necessary columns, such as "skill_id" (unique identifier) and "skill_name" (the name of the skill). Populate the "Skills" table with a predefined set of skills that users can choose from when filling out their profile or CV.
- Professions: Create a database table named "professions" to store a list of professions. Define columns, such as "profession_id" (unique identifier) and "profession_name" (the name of the profession). Populate the "professions" table with a predefined set of professions that users can select when specifying their work experience or career goals.
- Degrees: Create a database table named "degrees" to store a list of educational degrees or qualifications. Define columns, such as "degree_id" (unique identifier) and "degree_name" (the name of the degree). Populate the "degrees" table with a predefined set of degrees that users can choose from when specifying their educational background.
- Education: Create a database table named "education" to store a list of educational institutions. Define columns, such as "education_id" (unique identifier) and "education_name" (the name of the institution). Populate the "Education" table with a predefined set of educational institutions that users can select when entering their educational history.

Return Multiple Choice Lists: Implement functionality to retrieve the pre-filled data from the respective database tables and provide it as multiple-choice lists for the user.

For example, when a user fills out their profile or CV, they can select skills from the list of pre-defined skills, choose a profession from the list of available professions, select their degree from the pre-populated degrees, and choose their educational institution from the list of available educational institutions.

Create appropriate API endpoints or methods in your .NET application to fetch the data from the database tables and return it as response objects or JSON arrays that can be consumed by the user.

User Profile (Applicant Role):

User with the “Applicant” role can create CV with the personal information. A user creates his CV once and then the data can be changed by editing. You can also delete your own CV.

Filling out the CV by the User:

Implement a feature that allows users to create and edit their CV (Curriculum Vitae) information. Include the following data fields for the CV:

- Contact Information: first name, last name, phone number and email address.
- Education: Include the names of the institutions, dates of attendance, educational qualification and profession.
    1. Choose educational institution from the list of available educational institutions.
    2. Enter dates of attendance (start date, end date).
    3. Select degree from the pre-populated degrees list.
    4. Choose a profession from the list of available professions.
- Skills and Work Experience: Select skills from the list of pre-defined skills list and indicate work experience of each skill in years. (ex. .net: 5);

Implement functionality that allows users to save their CV data, edit it at a later time, and delete their own CV if desired.

Validate the input data to ensure that all required fields are filled in correctly and that the data is in the expected format.

Implement functionality that restricts users from accessing or modifying data belonging to other users. Only the authenticated user should be able to view and update their own profile information. the user can see only own data;

Applicants can submit applications for the presented jobs and apply to multiple positions using a single CV.

User Profile (Employer Role):

Employer Access to All Data. “Employer” role has the manager who creates the job description. Implement functionality that allows managers to access and view data for all job descriptions and job offers within the system.

Apply appropriate authorization mechanisms to ensure that only Employer have access to this privileged information.

Implement functionality that allows the employer to create job descriptions, implement a feature that enables managers to select the necessary skills from a pre-defined list.

- Job Title: Specify the title of the job.
- Requirements: Define the required skills from the list of pre-defined skills list and desired experience for the job in years (numeric value ex. 5). Its mandatory field and manager must indicate at least 5 skills for the job.
- Job Expiration date. Applicants can not apply to expired jobs.

Implement validation checks to ensure that all required fields are filled in before allowing the manager to create the job description.

Store the job description data in the database, associating it with a unique identifier for future reference and retrieval.

Valuable Skills Selection:

Extend the job offer creation functionality to allow managers to specify valuable skills required for each job offer. Allow managers to select and assign weights or priorities to each skill, indicating their importance for the job. Store the selected valuable skills and their corresponding weights in the database, associating them with the respective job offer.

Skill Matching Algorithm:

Implement an algorithm to calculate the match between the skills indicated by candidates in their CVs and the valuable skills specified by managers for each job offer.

Retrieve the valuable skills and their weights associated with a particular job offer from the database.

Retrieve the candidate's skills from their CV data.

Compare the skills listed in a candidate's CV with the valuable skills for the corresponding job offer specified by managers, taking into account the assigned weights or priorities for each skill.

Calculate a score indicating the level of skill match between the candidate's skills and the required valuable skills, considering the assigned weights or priorities.

Store the calculated match score along with the candidate's CV data, allowing managers to evaluate the suitability of candidates for the job.

Provide a feature for managers to view and analyze the recommended candidates for each job offer based on the skill matching algorithm. Present the candidates in descending order of their skill match score.

Job Offer Management:

Develop a functionality for managers to view and manage all job offers within the system.

Display a list of job offers, including relevant details such as job title, qualification, requirement, work experience and validity date information.

Implement functionality to delete and update job offers. Managers should be able to make changes to job offer details when appropriate.

Implement validation checks to ensure that only authorized managers can perform these actions on job offers.

Skill Matching Algorithm:

Implement an algorithm to calculate the match between the skills indicated by candidates in their CVs and the valuable skills specified by managers for each job offer.

Retrieve the valuable skills and their weights associated with a particular job offer from the database.

Retrieve the candidate's skills from their CV data. For each skill in the candidate's skills: a. Check if the skill is included in the valuable skills specified by managers for the job offer. b. If the skill is present:

Retrieve the weight assigned to that skill for the job offer.

Compare the candidate's skills with the valuable skills specified by managers, taking into account the assigned weights or priorities for each skill.

Calculate a skill match score or percentage indicating the level of skill match between the candidate's skills and the required valuable skills.

Store the calculated skill match score or percentage along with the candidate's CV data, allowing managers to evaluate the suitability of candidates for the job.

Best Candidate Recommendation:

Provide a feature that allows managers to view and analyze the recommended candidates for each job offer based on the skill matching algorithm.

Present the recommended candidates in descending order of their skill match score, allowing managers to identify the best-suited candidates for the job.

Include relevant candidate details such as name, contact information, work experience, and education in the recommendation view.

Implement filters or sorting options to allow managers to refine the candidate list based on specific criteria or preferences.

Additional Considerations:

Implement secure authentication and authorization mechanisms to ensure that only authorized managers can access and modify job offer and candidate-related data.

Design and implement the necessary database schema updates to store skill weights, skill match scores, and other relevant data.

Consider implementing additional features such as filtering, sorting, and searching capabilities within the job offer management dashboard to enhance usability.

Optimize the skill matching algorithm for efficiency, considering potential performance implications when dealing with a large number of candidates and job offers.

Provide appropriate error handling and validation checks to ensure the accuracy and integrity of data during skill matching and job offer management processes.

Test the functionality thoroughly, considering various scenarios, to ensure accurate skill matching calculations and proper management of job offers and candidate recommendations.